Group 6
Dennis Joseph Mayuga
Alex Magda
Chaohao Qiu

Final Project Document

I. PROJECT SCOPE

This restaurant management system is intended for use by waiters in real-time as they greet and serve customers at the table to track orders for all seven tables simultaneously. The restaurant management system also allows waiters to track their total sales and tables served. A restaurant administrator can access this information in the database for the restaurant's total sales and sales by a specific waiter to evaluate who is performing up to standard. Admins of this management system may also update the restaurant's official menu stored in the database, which all waiters/customers access and operate through. Currently, there is no distinction between types of accounts; the system accepts a waiter's username and password to log in and can access all of the available features.

II. PROJECT FUNCTIONALITIES

- Log in system (create, retrieve, authenticate matching information)
- Menu system (create, retrieve, update, delete)
- Sales history system (retrieve, update; delete with checkout)
- Table layout (create, retrieve, update. delete, persist information)

III. PROJECT SETUP

Versions: Final Submission Version

How to setup:

- Clone the master branch of the RestaurantManagementSytem repository.
- Create a new database in your local server and create tables with the information provided in the SQLTABLES.txt file of the repository.
- Change the connection strings in our code to your actual connection string in the following locations:
  - LoginAPI folder
    - ConnectionString.cs line 5
  - MenuAPI folder
    - appsettings.json line 9
    - ConnectionString.cs line 5

- SalesAPI folder
    - SalesService.cs line 9
- NOTE: If you are not certain to have gotten them all, press CTRL+SHIFT+F and write “data source” to see every instance of these words in the overall project and ensure that you have replaced all of them.

- Run all three APIs from their respective folders
    - LoginAPI folder: API.sln
    - MenuAPI folder: RMSAPI.sln
    - SalesAPI folder: SalesAPI.sln
- Run RMSWPF.sln in the RMSWPF folder.
-

IV. MIDTERM PROJECT DISCUSSION

At the time of the mid-term project discussion, the first prototype of the WPF was completed, and the MenuAPI was almost completed but not running due to build errors, meaning that the other three APIs at the time had not been started. By the presentation, about a quarter of the project was properly implemented, and by the final demo, around 90% of it was done.

The navigation buttons of the WPF were fully created, and the database tables were set up. The MenuAPI’s service functions were all written, but the update function was not retrieving the right information on the Swagger UI browser. As a result, the MenuAPI and WPF were not yet connected, so the buttons on the WPF that called these functions had no code. The MenuAPI’s create, retrieve, and delete functions were working.

At the time of the midterm discussion, we planned to add another API that would create, retrieve, update, and delete items from the customers’ orders in each individual table, as well as another API that would update and retrieve total sales per day into the Sales History window as a management function to track the restaurant’s profits and success.

V. WORK PROGRESS IN FUTURE

This restaurant management system is rudimentary and can be improved by adding different types of accounts (waiters and managers), which would then show different options; for example, the sales history being available only to manager accounts. A feature to input how many tables the restaurant has and automatically generate tables for that amount of tables would also be useful, as opposed to having a fixed number of seven tables only. Adding a feature to register how much the store made in tips would also be useful. A proper restaurant management system would also need a functioning point-of-sale system for checkout once the order is complete, as the service right now simply clears the current table, updates sales history, and redirects the user to the main navigation menu when the checkout button is pressed. Some exceptions are not handled gracefully, such as primary key violations, which terminate the WPF when it should be refined to give an error message and allow the user to retry for a valid input. The WPF could also be reorganized to look more aesthetically pleasing, as well as edit some features to make them inherently user-friendly. Rearranging the order that the WPF's buttons appear in so that pressing on tab will go to the next input field or button that appears visually would be very useful.

More detailed data from tableCart checkout could be sent to other tables for further data analysis(ie. Inventory management, market analysis etc.) or accounting purpose.

This restaurant management system could be used by any restaurant with a dine-in section. If the system is expanded to allow orders for pickup, such as UberEats, (meaning the addition of a website with the menu that would be updated automatically every time the menu is updated from the manager's account, as well as other features necessary).

VI. PARTICIPANTS CONTRIBUTION

Work distribution:

- Dennis:
  - LoginAPI
  - Debugged and fixed everyone’s code when help was requested
- Alex:
  - Base WPF creation
  - Prototype MenuAPI
  - SalesAPI
- Chaohao:
  - CartAPI
  - Edited and finalized WPF to match all APIs

On average, the team met for at least one evening after classes per week to work on this project. We put in a lot more time, almost every evening, after the midterm discussion in order to implement all the features we wanted to integrate.